NYU Tandon School of Engineering
CS-GY 6083, Principles of Database Systems, Fall 2024
Prof Phyllis Frankl

Practice Material #2 Solutions

Problem 1

a) (Car.carMake, Car.carModel, Car.carModelNum) is the foreign key reference to (CarType.carMake, CarType.carModel, Car.carModelNum)
Rental.cID is the foreign key reference to Customer.cID
Rental.carID is the foreign key reference to Car.carID
Rental.pickBid is the foreign key reference to Branch.bid
Rental.returnBid is the foreign key reference to Branch.bid

b) (1)
```
Select distinct Customer.cID, cName
From Customer Natural Join Rental Natural Join Car Natural Join
CarType
Where CarType.carModel = 'Toyota' and carColor = 'blue' and carSeats =
6 and year(pickupTD) = 2017
```

(2)
```
(Select distinct cID From Rental Natural Join Car Where carModel =
'Toyota')
Except
(Select distinct cID From Rental Natural Join Car Where carModel =
        'Audi)
```

(3)
```
Select carID, count(rID) as Times
From Rental
Group by carID
```

(4)
```
Create temporary table t1
Select count(distinct bState) as totalNumofStates
From Branch
Create temporary table t2
Select cID, count(distinct bState) as numofStates
From Rental Natural Join Branch
Group by cID
Select cID,
From t1, t2
Where numofStates = totalNumofStates
Drop t1, t2
```

(5)
```
Create temporary table t1
Select cID, sum(cost) as totalCost,
From Rental
Where year(pickupTD) = 2017
Group by cID
Create temporary table t2
Select max(totalCost) as maxCost
From t1
Select Customer.cID, cName
From t1, t2, Customer
Where Customer.cID = t1.cID and totalCost = maxCost
Drop table t1, t2
```

(6)
```
Create temporary table t1
Select max(cost) as maxCost
From Rental
Where year(pickupTD) = 2017
Select Customer.cID, cName
From t1, Rental, Customer
Where Customer.cID = Rental.cID and cost = maxCost
Drop t1
```

(c)

I.

$\{ res \mid \exists u \in Customer (res[cID] = u[cID] \wedge res[cName] = u[cName] \wedge r \in Rental(r[cID] = u[cID] \wedge r[pickupTD].year = 2017 \wedge \exists c \in Car (c[carID] = r[carID] \wedge c[carMake] = Toyota \wedge \exists t \in CarType(c[carMake] = t[carMake] \wedge c[carModel] = t[carModel] \wedge c[carModelNum] = t[carModelNum] \wedge t[carSeats] = 7)))) \}$

II.

$\{ res \mid \exists tr \in Rental (res[cID] = tr[cID] \wedge \exists t \in Car(t[carMake] = Toyota \wedge tr[carID] = t[carID])) \wedge \neg \exists ar \in Rental (res[cID] = ar[cID] \wedge \exists a \in Car(a[carMake] = Audi \wedge ar[carID] = a[carID])) \}$

III.

As TRC/DRC cannot express aggregation function, it cannot be expressed.

IV.
{ c | ∀b' ∈ Branch ⇒ ∃r ∈ Rental (c[cID] = r[cID] ∧ ∃b ∈ Branch (b'[bState] = b[bState] ∧ r[pickupBid] = b[bid])) }

V.
As TRC/DRC cannot express aggregation function, it cannot be expressed.

VI.
{ res | ∀r' ∈ Rental (r'[pickupTD].year = 2017 ⇒ ∃r ∈ Rental (res[cID] = r[cID] ∧ r[pickupTD].year = 2017 ∧ r[cost] ≥ r'[cost] ∧ ∃c ∈ Customer (r[cID] = c[cID] ∧ res[cName] = c[cName]))) }

Problem 2

(a)

- Customer (**cID**, cName, cPhone, cCard, ccID)
- Location (**lID**, name, street, city, zip, state, phone)
- Shipment (**sID**, sDateTime, weight, sourceID, destinationID, cost, senderID, payerID)
- Track (**tID**, sID, tDateTime, tLongitude, tLatitude, tCity, tZip, tDescription)
- ZipCategory (**zip**, category)
- Price (minWeight, **maxWeight, sourceCategory, destinationCategory, isInSameState**, pPrice)
- CustomerClass (**ccID**, className, discount)

Foreign Key:

- Customer (ccID) -> CustomerClass (ccID)
- Location (zip) -> ZipCategory (zip)
- Shipment (sourceID) -> Location (lID)
- Shipment (destinationID) -> Location (lID)
- Shipment (senderID) -> Customer (cID)
- Shipment (payerID) -> Customer (cID)
- Track (sID) -> Shipment (sID)

(b)

I.

```
select cID, cName, sum(cost)
from Customer join Shipment on cID = payerID
where getYear(sDateTime) = 2017
group by cID, cName
```

II.

```
with TrackLastTime as
(select sID, tDateTime as tDateTimeLast
from Track
where tDescription = “Arrival scan, Miami airport” and now() – tDateTime > 5 * 24 *
3600);

select sID
from TrackLastTime
where sID not in (
select sID
from TrackLastTime natural join Track
where tDateTime > tDateTimeLast
)
```

Problem 3

a.

1.

```
select eid
from Club
   natural join HoldsEvent
   natural join Event
where
   cname = "French Literature Club"
   and maxpeople > 10;
```

2.

```
select sname, cid
from
   Student
   natural join Register
   natural join HoldsEvent
   natural join Event
where
   edate.year = 2023
   and price < 10;
```

3.

Up to Assignment 2, you are allowed to write answers like:

```
select sname, eid
from Student
   natural join Register
   natural join Event
where
   price = nonmemprice;
```

– **But after hw3, following answers are preferred** :

```
select sname, eid
from Student
   natural join Register
   natural join Event
where
        sid NOT IN (SELECT sid FROM Members NATURAL JOIN Holds WHERE eid = e.eid)
```

4.

```
select sname, cid
from Student
natural join Membership
where
   year = 2022
   and semester = "Fall"
   and memberfee < 20;
```

b.

```
create table Student (
sid integer auto_increment primary key,
sname varchar(50),
semail varchar(50),
sphone varchar(50)
);

create table Club(
cid integer auto_increment primary key,
cname varchar(50),
cdescription varchar(50)
);
create table Event(
eid int auto_increment primary key,
ename varchar(50),
edescription varchar(50),
```

```
edate datetime,
memprice int,
nonmemprice int,
maxpeople int
);


create table Membership(
sid int,
cid int,
semester varchar(50),
year int,
memberfee int,
primary key(sid,cid,semester,year),
FOREIGN KEY (sid) REFERENCES Student(sid) ON DELETE CASCADE,
FOREIGN KEY (cid) REFERENCES Club(cid) ON DELETE CASCADE
);
create table HoldsEvent(
eid int,
cid int,
primary key(eid ,cid),
FOREIGN KEY (eid) REFERENCES Event(eid) ON DELETE CASCADE,
FOREIGN KEY (cid) REFERENCES Club(cid) ON DELETE CASCADE
);
create table Register(
eid int,
sid int,
price int,
rating float,
primary key(eid ,sid),
FOREIGN KEY (eid) REFERENCES Event(eid) ON DELETE CASCADE,
FOREIGN KEY (sid) REFERENCES Student(sid) ON DELETE CASCADE
);
```

c.

1.

$$\Pi_{eid}(\sigma_{cname=\text{"FrenchLiteratureClub"} \wedge maxpeople>10}(Club \bowtie HoldsEvent \bowtie Event))$$

2.

$$\Pi_{sname,cid}(\sigma_{edate.year=2023 \wedge price<10}(Student \bowtie Register \bowtie HoldsEvent \bowtie Event))$$

3.

$$\Pi_{sname,eid}(\sigma_{price=nonmemberprice}(Student \bowtie Register \bowtie Event))$$

4.

$$\Pi_{sname,cid}(\sigma_{year=2022 \wedge semester=\text{"}Fall\text{"} \wedge memberfee<20}(Student \bowtie Membership))$$

d.

1.

$$\{< ei > | \exists en, ed, eda, em, en, ema(< ei, en, ed, eda, em, en, ema > \in Event \\ \wedge \exists ci(< ei, ci > \in HoldsEvent \\ \wedge \exists cn, cd(< ci, cn, cd > \in Club \\ \wedge cn = \text{"}French\ Literature\ Club\text{"} \wedge ema > 10)))\}$$

2.

$$\{< sn, ci > | \exists si, se, sp(< si, sn, se, sp > \in Student \\ \wedge \exists ei, p, r(< ei, si, p, r > \in Register \\ \wedge (< ei, ci > \in HoldsEvent \\ \wedge \exists en, ed, eda, em, en, ema(< ei, en, ed, eda, em, en, ema > \in Event \\ \wedge ed.year = 2023 \wedge p < 10))))\}$$

3.

$$\{< sn, ei > | \exists si, se, sp(< si, sn, se, sp > \in Student \\ \wedge \exists p, r(< ei, si, p, r > \in Register \\ \wedge \exists en, ed, eda, em, en, ema(< ei, en, ed, eda, em, en, ema > \in Event \\ \wedge p = en)))\}$$

4.

$$\{< sn, ci > | \exists si, se, sp(< si, sn, se, sp > \in Student \\ \wedge \exists s, y, m(< si, ci, s, y, m > \in Membership \\ \wedge y = 2022 \wedge s = \text{"}Fall\text{"} \wedge m < 20)))\}$$

Problem 4

1.

```
select RideTitle from Locate where ParkTitle = 'MagicKingdomPark' and Addr = 'Bay Lake';
```

2.

```
select tid, first_name, last_name from Tourist where year(DOB) = 2001;
```

3.

```
select first_name, last_name, Email from Visit natural join Tourist where ParkTitle =
'Disney''sHollywoodStudios' and Addr = 'Bay Lake';
```

4.

```
select  first_name, last_name, Avg_wt  from Tourist natural join favorite natural join ride
where Avg_wt>"00:30:00";
```

5.

```
select first_name, last_name from Tourist natural join Favorite natural join locate natural join
visit where stars <> 5;
```